Prettyprinting Styles for Various Languages

Gary T. Leavens
Massachusetts Institute of Technology
Laboratory for Computer Science
545 Technology Square, Cambridge, Mass. 02139

ABSTRACT

There are three basic styles of prettyprinting in current usage. I present a simple hypothesis that governs which style is commonly found for any given language, based on the observation of several programming languages.

1. Background

Like most programmers, I have a preference for a particular style of prettyprinting. Unfortunately, some of my colleagues have adopted what seemed (to me) a very strange style. I thought I could make them see the "error of their ways" if I could show them that *no one* uses their style of prettyprinting. However, I found that quite a number of programmers, and in fact whole language communities, seem to use their particular style of prettyprinting. In the course of this small bit of research, I discovered some simple rules that seem to govern what is acceptable prettyprinting style for a programming language.

I have come to view this diversity of prettyprinting styles as a linguistic phenomenon, and thus I can make no (objective) arguments as to why one style should be better than another. Of course, I have my own (subjective) reasons for preferring my favorite style, but they are not of interest here.

2. The Styles

There are three styles commonly found for matching a closing delimiter (for instance **end** or ')') with an opening delimiter. Other styles are possible, but seldom if ever used.

2.1. Ada Style

The Ada reference manual [10] uses and encourages the following prettyprinting style.

```
keyw
    body
end keyw
```

That is, the **end keyw** keyword is aligned directly under the keyword, and the body is indented. The amount of indentation of the body is not important, as long as it is definitely indented.

2.2. PL/I Style

PL/I [3, 9] programs are often (but not always) prettyprinted with the following style.

```
keyw
    body
    end
```

Again the body is indented, but note that the **end** keyword is aligned with the body, and not with the keyword (often **do**) that begins the structured statement.

2.3. Lisp Style

Lisp [19] prettyprinters use the following style.

```
(key
    body)
```

That is, the closing parenthesis is not given a separate line, but is placed on the end of the last line of the body that it closes.

2.4. Other Possible Styles

If one thinks about it, there is no *a priori* reason to rule out other possible styles. For instance, indenting the closing keyword even farther than the body.

```
keyw
    body
        end
```

Or indenting the closing keyword halfway between the indentation of the body and the indentation of the opening keyword.

```
keyw
        body
    end
```

Or not indenting the body or the closing keyword at all.

```
keyw
body
end
```

Note that all the styles commonly in use for languages with structured statements indent the body of a structured construct. Thus one may surmise that one will never see a prettyprinting style that does not indent the bodies of structured statements. (Languages like BASIC [4] and SNOBOL [13] do not have structured statements to indent.)

These alternative styles seem rarely, if ever to be used for languages with structured constructs. Probably they make visual pattern recognition of nesting levels difficult.

3. Discussion

When I first started to look at prettyprinting styles, I decided to survey whatever books I had that contained examples of formatted programs. Because I have access to quite a few programming language manuals and references, I was able to sample a large number of languages. It was during this process that I started to observe the three classes of prettyprinting styles shown above. Part of my survey apears below.

Ada Style	PL/I Style	LISP Style
Ada [10]	PL/I [9]	LISP [22]
Pascal [15]	CSP/k [14]	BCPL [20]
Edison [7]	CLU [18]	ML [12]
Modula-2 [23]		COBOL [2]
Algol 68 [21]		Prolog [8]
Concurrent Pascal [6]		MDL [11]

One should not take the sheer number of languages in any one category as some indication of "goodness".

Based on this data I made the following hypotheses.

1. If a language has a uniform, and small (typically single character) closing keyword which is also a punctuation symbol (such as ')' in LISP or '.' in COBOL and Prolog), then programmers tends to use LISP style prettyprinting.

2. If a language has a uniform, but rather larger or more visually arresting closing keyword (such as **end** in PL/I and CLU), then PL/I style is preferred.

3. If a language has several distinct closing keywords (such as **esac** and **fi** in Algol 68) or "end x" combinations (such as **end if** in Ada), then Ada style is preferred.

These hypotheses work fairly well in some cases. For instance, the language VAL [1] has a syntax similar to CLU's except that it has separate "end_x" keywords for all its structured statements. The theory predicts prettyprinters will use Ada style for VAL, and indeed this is the case in [1]. Similarly, all LISP variants and SmallTalk (which uses ']' in some statements) tend to be prettyprinted using LISP style.

A striking example of the theory's correct prediction occurs for Russell [5]. This language has both differentiated end keywords (such as **in** ... **ni**) and a common terminator used in may cases ('{'...'}'). The hypotheses predict that **ni** is lined up with **in** as in Ada style, and '}' is simply tacked on the end. This is in fact the case in several Russell programs that I have seen in [5].

However, there are some problems with the theory. One interesting counter-example is the 'C' language. Kerhenihagn and Ritche [16] format using the Ada style, and I have seen several programmers use PL/I style. C has a short, universal delimiter for its structured statements, namely '}', so the theory would predict that it would be prettyprinted using LISP

style; but this is almost never the case.

Another problem is that some authors seem to advocate the same style for every language. For instance, Ledgard [17] advocates Ada style for PL/I code.

Perhaps there are some factors that the theory does not take into account, such as the legacy of the language (which languages it resembles or was based on), or the personal preferences of influential language designers or authors.

4. Conclusions

There are three basic prettyprinting styles used for programming languages. Some regular patterns seem to emerge which allow one to predict from the syntax of the language, how one will commonly see it indented. It is thus quite impossible to label one prettyprinting style as "better" than another.

5. Acknowledgements

I would like to thank Barbara Liskov for her suggestions and comments on an earlier draft.

6. REFERENCES

1. Ackerman, W. B. and Dennis, J. B. VAL -- A Value-Oriented Algorithmic Language: Preliminary Reference Manual. Tech. Rep. TR-218, MIT Lab. for Computer Science, June, 1979.

2. *American National Standard Programming Language COBOL.* American National Standards Institute, New York, N.Y., 1974. ANS X3.23-1974.

3. *American National Standard Programming Language PL/I.* American National Standards Institute, New York, N.Y., 1976. ANS X3.53-1976.

4. *American National Standard for Minimal BASIC.* American National Standards Institute, New York, N.Y., 1978. ANS X3.60-1978.

5. Boehm, H., Demers, A., and Donahue, J. An Informal Description of Russell. Tech. Rep. TR 80-430, Department of Computer Science, Cornell University, Oct., 1980.

6. Brinch Hansen, P. *The Architecture of Concurrent Programs.* Prentice Hall, Inc., Englewood Cliffs, N.J., 1977.

7. Brinch Hansen, P. Edison Programs. *Software—Practice and Experience 11* (1981), 397-414.

8. Clocksin, W. and Mellish, C. *Programming in Prolog.* Springer-Verlag, New York, N.Y., 1981.

9. Conway, R. and Gries, D. *An Introduction to Programming. A Structured Approach using PL/I and PL/C.* Winthrop Publishers, Cambridge, Mass., 1973.

10. United States Department of Defense. Reference Manual for the Ada Programming Language. *AdaTEC* (July 1982). Draft Revised MIL-STD 181.5

11. Dornbrook, M. and Blank, M. The MDL Programming Language Primer. MIT Lab. for Computer Science, 1981.

12. Gordon, M., Milner, A., and Wadsworth, C. *Edinburgh LCF, Lecture Notes in Computer Science, 78.* Springer-Verlag, New York, N.Y., 1979.

13. Griswold, R. E., Poage, J. F., and Polonsky, I. P. *The SNOBOL4 Programming Language (second edition).* Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

14. Holt, R., Graham, G., Lazowski, E., and Scott, M. *Structured Concurrent Programming with Operating Systems Applications.* Addison-Wesley, Reading, Mass., 1978.

15. Jensen, K. and Wirth, N. *PASCAL User Manual and Report.* Springer-Verlag, 1974.

16. Kernighan, Brian W. *The C Programming Language.* Prentice-Hall, Englewood Cliffs, N.J., 1978.

17. Ledgard, H. *Programming Proverbs.* Hayden Book Co., Rochelle Park, N.J., 1975.

18. Liskov, B., Atkinson, R., Bloom, T., Moss, E., Schaffert, C., Scheifler, B., and Snyder, A. *CLU Reference Manual, Lecture Notes in Computer Science, 114.* Springer-Verlag, New York, N.Y., 1981. Also MIT/LCS TR-225

19. McCarthy, J., *et al. LISP 1.5 Programmer's Manual.* MIT Press, Cambridge, Mass., 1965.

20. Richards, M., Evans, A., and Mabee, R. The BCPL Reference Manual. Tech. Rep. TR-141, MIT Lab. for Computer Science, Dec., 1974.

21. van Wijngaarden, A., Mailloux, B., Peck, J., Koster, C., Sintzoff, M., Lindsey, C., Meertens, L., and Fisker, R. Revised Report on the Algorithmic Language ALGOL 68. *SIGPLAN Notices 12*, 5 (1977), 1-70. This has also been published by Springer-Verlag, New York, 1976

22. Winston, P., and Horn, B. *LISP.* Addison-Wesley, Reading, Mass., 1981.

23. Wirth, N. *Programming in Modula-2.* Springer-Verlag, New York, N.Y., 1982.